Welcome to Animal Crossing

(GCN)

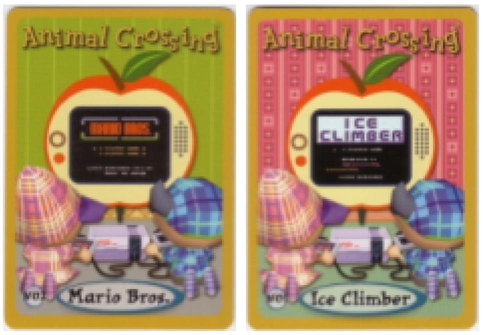
e-Reader Cards & Dot Code Documentation

Obtaining Super Mario Bros and Legend of Zelda NES Games via e-Reader Dot Codes

by Hunter Rafferty

History

In the original Animal Crossing for the GameCube there are special furniture items that feature fully functional NES games that you can play in your house. Most of these items are obtainable through normal means in the game, such as buying them from Nook's or finding them in a villager's buried treasure game. However, there are four specific NES items that are not obtainable through normal means. Dubbed "the forbidden four," these games are Ice Climber, Mario Bros., Super Mario Bros, and The Legend of Zelda. These items do not appear in any shop, raffle, or event in the game and are specifically whitelisted from being obtained via Tom Nook's secret codes. Even though we can theoretically generate secret codes to obtain these items, entering the codes does not work due to this whitelist. On May 12th, 2003, Nintendo released Series 4 of the e-Reader card packs and made two of these games obtainable by scanning two specific cards at the Post Office via the e-Reader peripheral and a linked Gameboy Advance. These cards were N01 Ice Climber and N02 Mario Bros. and are the only ways possible to obtain these items legitimately in-game.



N01 Ice Climber and N02 Mario Bros. e-Reader cards

Since Nintendo released these two cards, it was thought they would release the other cards to unlock Super Mario Bros and The Legend of Zelda. However, due to the commercial failure of the e-Reader in North America, these cards were never printed. Due to this, it's long been theorized that a certain e-Reader dot code must possibly exist to scan in-game and finally get these items, even if Nintendo never printed them. Many have tried to replicate the Mario Bros. and Ice Climber cards, but to my knowledge, this has never been achieved on console. That is, until now. Over 20 years later and after some painstaking reverse-engineering, I have created a working e-Reader dot code card that can be printed and scanned in-game to give you both items legitimately for the first time. **This can be done on a vanilla console with no hacks and no action replay** for the very first time – simply print the card using a printer and scan using a North American e-Reader.

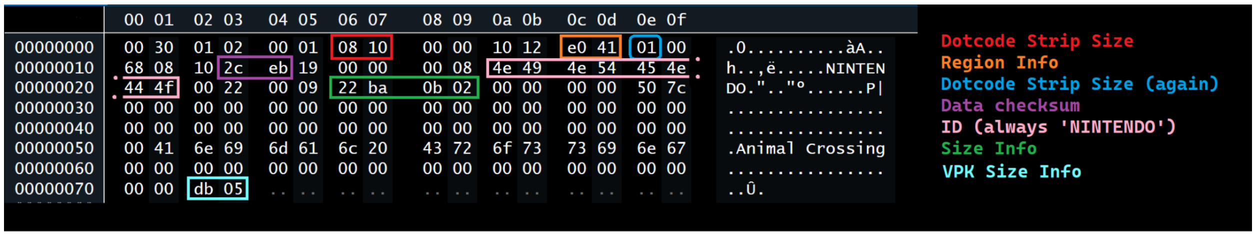
Explanation

This is possible by taking the existing Animal Crossing e-Reader cards, extracting their dot code data, and modifying certain flags to create a new dot code with new data. More specifically, I decompiled existing e-Reader card data as a base, made modifications to the data, and recompiled it replicating Nintendo's (more specifically HAL's) compression methods to make a new working card. This took quite a bit of research and time to accomplish, especially since e-Reader documentation and development is extremely dated. That said, I hope to detail and document my process further for any future developers interested in toying with the e-Reader or dot codes.

Technical Details

All e-Reader cards start as `.raw` files that can then be converted into `.bin` files for viewing in a hex editor. As a `.bin` file, the Animal Crossing cards that can be scanned in the Post Office are split into three chunks – one containing header data, one containing `.vpk` data for the Gameboy Advance, one one containing `.vpk` data for the GameCube.

The header data contains general information and checksums, including the dot code size, the size of the GBA VPK data, etc. Below is a graphic detailing the header data in a hex editor.



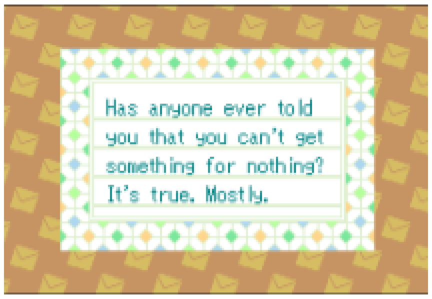
```
          00 01  02 03  04 05  06 07   08 09  0a 0b  0c 0d  0e 0f
00000000  00 30  01 02  00 01  08 10   00 00  10 12  e0 41  01 00   .0.........àA..
00000010  68 08  10 2c  eb 19  00 00   00 08  4e 49  4e 54  45 4e   h..,ë.....NINTEN
00000020  44 4f  00 22  00 09  22 ba   0b 02  00 00  00 00  50 7c   DO."..".º.....P|
00000030  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000040  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000050  00 41  6e 69  6d 61  6c 20   43 72  6f 73  73 69  6e 67   .Animal Crossing
00000060  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000070  00 00  db 05  .. ..  .. ..   .. ..  .. ..  .. ..  .. ..   ..Û.
```

Dotcode Strip Size
Region Info
Dotcode Strip Size (again)
Data checksum
ID (always 'NINTENDO')
Size Info
VPK Size Info

Header data for the Mario Bros. Series 4 e-Reader Card

When modifying e-Reader cards, this header is crucial to keep consistent. If the strip size, region, general size, VPK size, or any of the checksums are incorrect then the card will give a 'READ ERROR'. We will revisit this when going over how I actually modified the cards.

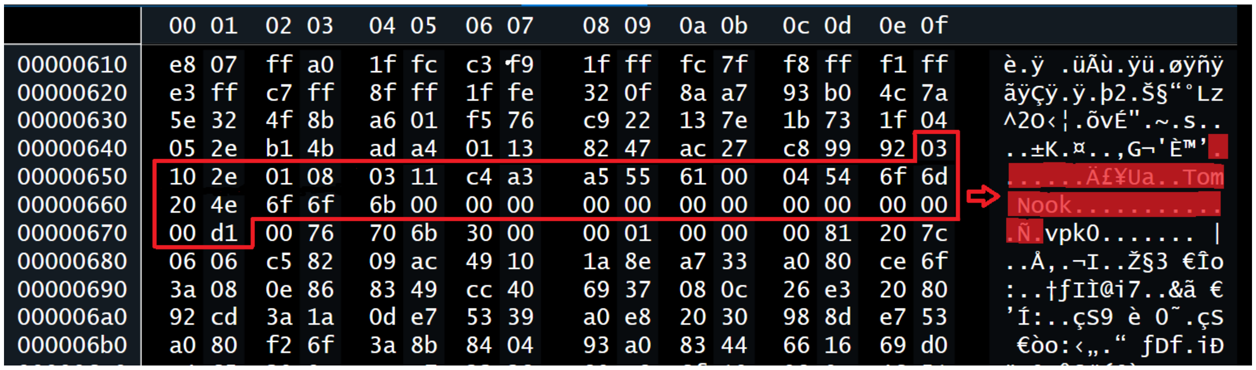
The next chunk in these cards contains VPK data specifically for the Gameboy Advance; that is, this data gets read by the GBA when you simply scan the card in the e-Reader BIOS (*not* in the Animal Crossing e-Reader application you download from the GameCube). Most Animal Crossing e-Reader cards have this data to include a letter that shows up on the GBA when scanned, accompanied by a special background and character music.



GBA eReader letter from Ice Climber Series 4 card

This data is not actually important when modifying the cards to get in-game items for Animal Crossing, but it is required to be there for the card to function. However, the text that appears on these cards is fully editable in a hex editor once you decompile the VPK, letting you write custom letters that appear on the GBA if interested. I have not tested this extensively, but there appears to be a 256 character (in bytes) limit to this text. Going past this amount is not recommended. There is also data within the VPK that controls the graphics, background, and music that play during this letter screen, but changing those is more complicated than it's worth.

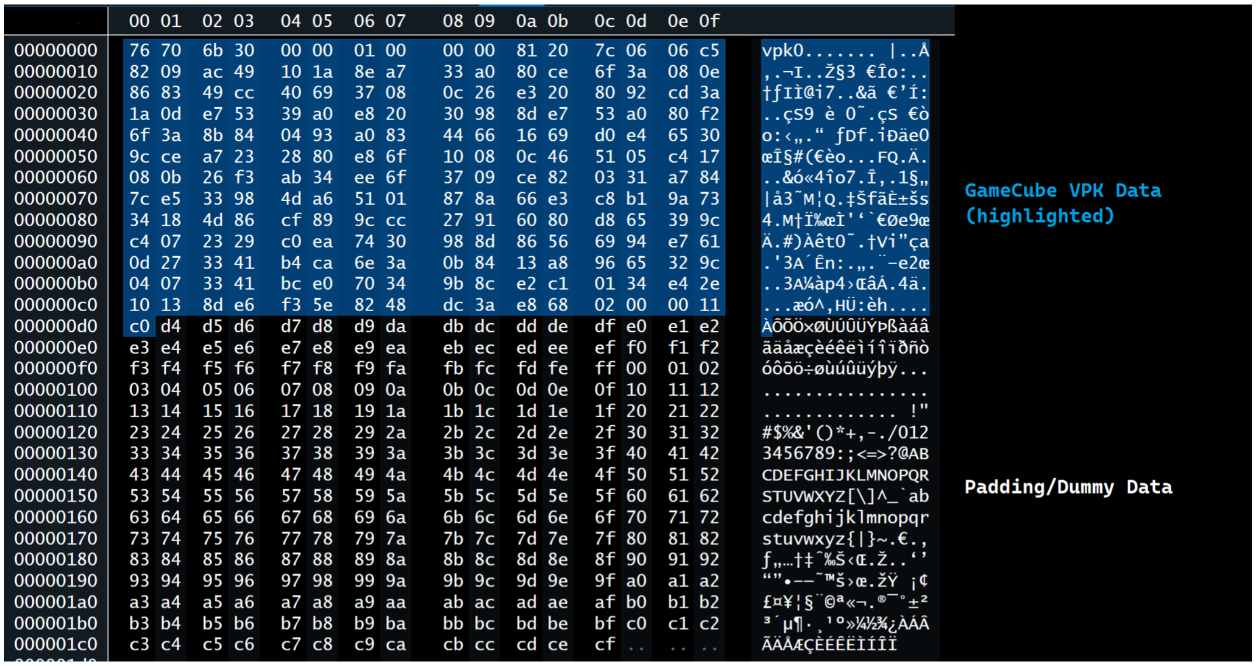
The important part of the GBA VPK data comes from the padding that occurs after the initial VPK data, but before the start of the GameCube VPK data. This data controls the "letter item" data that you receive in the mail when you scan the card in Animal Crossing. For the sake of receiving NES games as a present in the letter, this letter data **must remain the exact same**. In the example below you can see the letter is programmed to be from Tom Nook.



```
          00 01  02 03  04 05  06 07   08 09  0a 0b  0c 0d  0e 0f
00000610  e8 07  ff a0  1f fc  c3 f9   1f ff  fc 7f  f8 ff  f1 ff    è.ÿ .üÃù.ÿü.øÿñÿ
00000620  e3 ff  c7 ff  8f ff  1f fe   32 0f  8a a7  93 b0  4c 7a    ãÿÇÿ.ÿ.þ2.Š§“°Lz
00000630  5e 32  4f 8b  a6 01  f5 76   c9 22  13 7e  1b 73  1f 04    ^2O‹¦.õvÉ".~.s..
00000640  05 2e  b1 4b  ad a4  01 13   82 47  ac 27  c8 99  92 03    ..±K.¤..‚G¬'È™’.
00000650  10 2e  01 08  03 11  c4 a3   a5 55  61 00  04 54  6f 6d    ......Ä£¥Ua..Tom
00000660  20 4e  6f 6f  6b 00  00 00   00 00  00 00  00 00  00 00     Nook...........
00000670  00 d1  00 76  70 6b  30 00   00 01  00 00  00 81  20 7c    .Ñ.vpk0....... |
00000680  06 06  c5 82  09 ac  49 10   1a 8e  a7 33  a0 80  ce 6f    ..Å‚.¬I..Ž§3 €Îo
00000690  3a 08  0e 86  83 49  cc 40   69 37  08 0c  26 e3  20 80    :..†ƒIÌ@i7..&ã €
000006a0  92 cd  3a 1a  0d e7  53 39   a0 e8  20 30  98 8d  e7 53    ’Í:..çS9 è 0˜.çS
000006b0  a0 80  f2 6f  3a 8b  84 04   93 a0  83 44  66 16  69 d0     €òo:‹„.“ ƒDf.iÐ
```

Letter item data from the Mario Bros. Series 4 e-Reader card

The final chunk of data is the GameCube VPK data. Different from the letter item data, this data controls the letter contents (what is actually written in the letter) as well as what item present is attached. Both of these are editable – so you can receive a letter with customized text as well as most items as an attached present. These must remain within the constraints of the game, however, so no going beyond the letter text limit and no attaching items that cannot normally be attached to letters.



```
          00 01  02 03  04 05  06 07   08 09  0a 0b  0c 0d  0e 0f
00000000  76 70  6b 30  00 00  01 00   00 00  81 20  7c 06  06 c5    vpk0....... |..Å
00000010  82 09  ac 49  10 1a  8e a7   33 a0  80 ce  6f 3a  08 0e    ‚.¬I..Ž§3 €Îo:..
00000020  86 83  49 cc  40 69  37 08   0c 26  e3 20  80 92  cd 3a    †ƒIÌ@i7..&ã €’Í:
00000030  1a 0d  e7 53  39 a0  e8 20   30 98  8d e7  53 a0  80 f2    ..çS9 è 0˜.çS €ò
00000040  6f 3a  8b 84  04 93  a0 83   44 66  16 69  d0 e4  65 30    o:‹„.“ ƒDf.iÐäe0
00000050  9c ce  a7 23  28 80  e8 6f   10 08  0c 46  51 05  c4 17    œÎ§#(€èo...FQ.Ä.
00000060  08 0b  26 f3  ab 34  ee 6f   37 09  ce 82  03 31  a7 84    ..&ó«4îo7.Î‚.1§„
00000070  7c e5  33 98  4d a6  51 01   87 8a  66 e3  c8 b1  9a 73    |å3˜M¦Q.‡ŠfãÈ±šs
00000080  34 18  4d 86  cf 89  9c cc   27 91  60 80  d8 65  39 9c    4.M†Ï‰œÌ'‘`€Øe9œ
00000090  c4 07  23 29  c0 ea  74 30   98 8d  86 56  69 94  e7 61    Ä.#)Àêt0˜.†Vi”ça
000000a0  0d 27  33 41  b4 ca  6e 3a   0b 84  13 a8  96 65  32 9c    .'3A´Ên:.„.¨–e2œ
000000b0  04 07  33 41  bc e0  70 34   9b 8c  e2 c1  01 34  e4 2e    ..3A¼àp4›ŒâÁ.4ä.
000000c0  10 13  8d e6  f3 5e  82 48   dc 3a  e8 68  02 00  00 11    ...æó^‚HÜ:èh....
000000d0  c0 d4  d5 d6  d7 d8  d9 da   db dc  dd de  df e0  e1 e2    ÀÔÕÖ×ØÙÚÛÜÝÞßàáâ
000000e0  e3 e4  e5 e6  e7 e8  e9 ea   eb ec  ed ee  ef f0  f1 f2    ãäåæçèéêëìíîïðñò
000000f0  f3 f4  f5 f6  f7 f8  f9 fa   fb fc  fd fe  ff 00  01 02    óôõö÷øùúûüýþÿ...
00000100  03 04  05 06  07 08  09 0a   0b 0c  0d 0e  0f 10  11 12    ................
00000110  13 14  15 16  17 18  19 1a   1b 1c  1d 1e  1f 20  21 22    ............. !"
00000120  23 24  25 26  27 28  29 2a   2b 2c  2d 2e  2f 30  31 32    #$%&'()*+,-./012
00000130  33 34  35 36  37 38  39 3a   3b 3c  3d 3e  3f 40  41 42    3456789:;<=>?@AB
00000140  43 44  45 46  47 48  49 4a   4b 4c  4d 4e  4f 50  51 52    CDEFGHIJKLMNOPQR
00000150  53 54  55 56  57 58  59 5a   5b 5c  5d 5e  5f 60  61 62    STUVWXYZ[\]^_`ab
00000160  63 64  65 66  67 68  69 6a   6b 6c  6d 6e  6f 70  71 72    cdefghijklmnopqr
00000170  73 74  75 76  77 78  79 7a   7b 7c  7d 7e  7f 80  81 82    stuvwxyz{|}~.€.‚
00000180  83 84  85 86  87 88  89 8a   8b 8c  8d 8e  8f 90  91 92    ƒ„…†‡ˆ‰Š‹ŒŽ..‘’
00000190  93 94  95 96  97 98  99 9a   9b 9c  9d 9e  9f a0  a1 a2    “”•–—˜™š›œ.žŸ ¡¢
000001a0  a3 a4  a5 a6  a7 a8  a9 aa   ab ac  ad ae  af b0  b1 b2    £¤¥¦§¨©ª«¬.®¯°±²
000001b0  b3 b4  b5 b6  b7 b8  b9 ba   bb bc  bd be  bf c0  c1 c2    ³´µ¶·¸¹º»¼½¾¿ÀÁÂ
000001c0  c3 c4  c5 c6  c7 c8  c9 ca   cb cc  cd ce  cf ..  .. ..    ÃÄÅÆÇÈÉÊËÌÍÎÏ
```

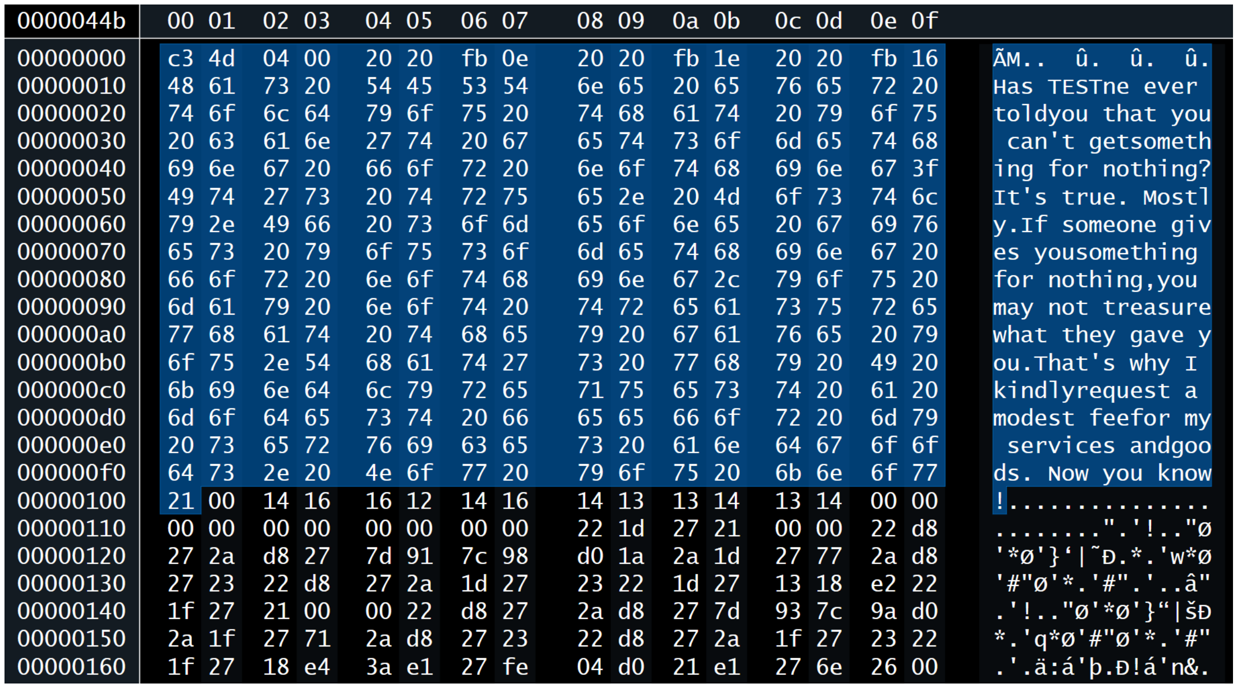
GameCube VPK data from the Mario Bros. Series 4 e-Reader card

Technical Process

*For a **tutorial** on how to actually modify Animal Crossing e-Reader card data, see my guide here.*

Obtaining Animal Crossing e-Reader cards in RAW format is quite simple following a few Google searches. However, it is a lot more difficult than simply opening those files and changing item code data. e-Reader cards come in a compressed state, meaning their data is mostly gibberish to the human eye unless we decompile them. Luckily, two people named CaitSith and Tim Schuerewegen took an interest in decompiling the e-Reader from 2004 to 2007. They created various tools that decompile/recompile e-Reader card data that allows them to be read and modified by eye. Their work and programs are published on CaitSith's site which is still live today. I cannot understate the importance of these programs – they laid the foundation of this discovery nearly 20 years later, so huge props to them. Further, custom Python code created by GameFAQs user RollingStar/do_0m was also instrumental in automating the decompiling process. While this code was unfinished, I was able to write further code and patch a few bugs to complete their original intention.

With these programs in hand, I was able to decompile the e-Reader cards from RAW to BIN to VPK and finally down to its bare source file. In these source files you can actually read the data in a hex editor and make modifications, so I was off to the races figuring out where I could make changes.

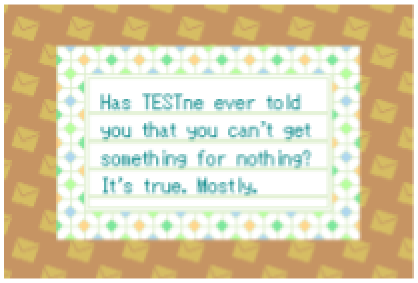


```
0000044b   00 01  02 03   04 05  06 07    08 09  0a 0b   0c 0d  0e 0f

00000000   c3 4d  04 00   20 20  fb 0e    20 20  fb 1e   20 20  fb 16    ÃM..  û.  û.  û.
00000010   48 61  73 20   54 45  53 54    6e 65  20 65   76 65  72 20    Has TESTne ever 
00000020   74 6f  6c 64   79 6f  75 20    74 68  61 74   20 79  6f 75    toldyou that you
00000030   20 63  61 6e   27 74  20 67    65 74  73 6f   6d 65  74 68     can't getsometh
00000040   69 6e  67 20   66 6f  72 20    6e 6f  74 68   69 6e  67 3f    ing for nothing?
00000050   49 74  27 73   20 74  72 75    65 2e  20 4d   6f 73  74 6c    It's true. Mostl
00000060   79 2e  49 66   20 73  6f 6d    65 6f  6e 65   20 67  69 76    y.If someone giv
00000070   65 73  20 79   6f 75  73 6f    6d 65  74 68   69 6e  67 20    es yousomething 
00000080   66 6f  72 20   6e 6f  74 68    69 6e  67 2c   79 6f  75 20    for nothing,you 
00000090   6d 61  79 20   6e 6f  74 20    74 72  65 61   73 75  72 65    may not treasure
000000a0   77 68  61 74   20 74  68 65    79 20  67 61   76 65  20 79    what they gave y
000000b0   6f 75  2e 54   68 61  74 27    73 20  77 68   79 20  49 20    ou.That's why I 
000000c0   6b 69  6e 64   6c 79  72 65    71 75  65 73   74 20  61 20    kindlyrequest a 
000000d0   6d 6f  64 65   73 74  20 66    65 65  66 6f   72 20  6d 79    modest feefor my
000000e0   20 73  65 72   76 69  63 65    73 20  61 6e   64 67  6f 6f     services andgoo
000000f0   64 73  2e 20   4e 6f  77 20    79 6f  75 20   6b 6e  6f 77    ds. Now you know
00000100   21 00  14 16   16 12  14 16    14 13  13 14   13 14  00 00    !...............
00000110   00 00  00 00   00 00  00 00    22 1d  27 21   00 00  22 d8    ........".'!.."ø
00000120   27 2a  d8 27   7d 91  7c 98    d0 1a  2a 1d   27 77  2a d8    '*ø'}'|˜Ð.*.'w*ø
00000130   27 23  22 d8   27 2a  1d 27    23 22  1d 27   13 18  e2 22    '#"ø'*.'#".'..â"
00000140   1f 27  21 00   00 22  d8 27    2a d8  27 7d   93 7c  9a d0    .'!.."ø'*ø'}"|šÐ
00000150   2a 1f  27 71   2a d8  27 23    22 d8  27 2a   1f 27  23 22    *.'q*ø'#"ø'*.'#"
00000160   1f 27  18 e4   3a e1  27 fe    04 d0  21 e1   27 6e  26 00    .'.ä:á'þ.Ð!á'n&.
```

Decompiled GBA VPK from Ice Climber Series 4 card with editable text that can be read and modified by hand

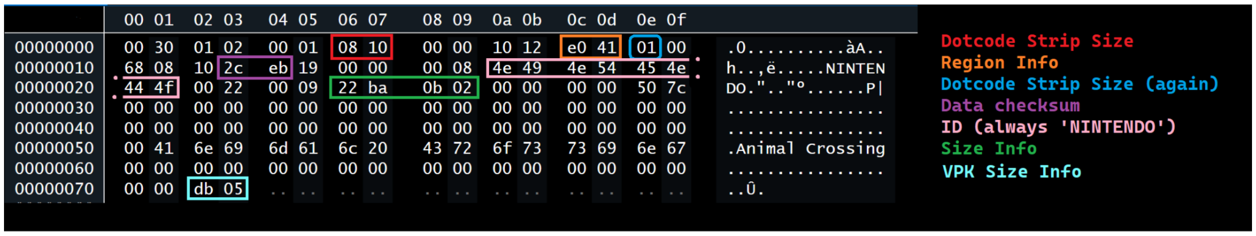
As you can see from this decompiled VPK, you can actually read and edit the bytes manually to change their data. In the above example, I inserted a random "TEST" text into the code and was eventually able to replicate the text in the GBA e-Reader.



GBA e-Reader letter featuring custom "TEST" text

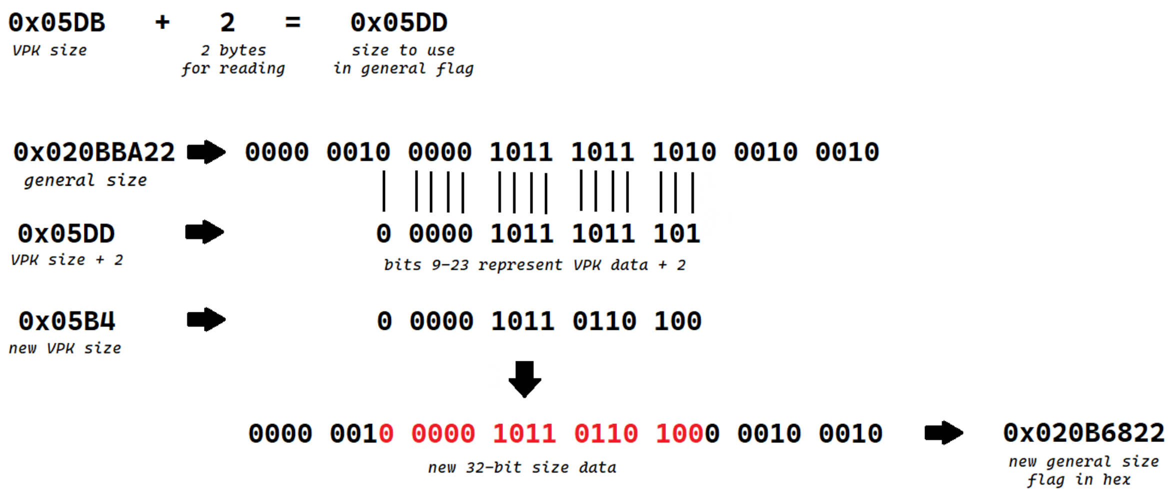
However, as mentioned, the process is not as simple as just modifying the data and recompiling. The tools used were great at *decompiling* data, but were actually **too** efficient at *recompiling* data. If you recall, the header data in each e-Reader card have "general size" and "VPK size" flags that must be correct in order for the card to not give an error. That means any change in VPK size or card size must be accounted for in the header. Thus, if the recompiler is too efficient at compressing the data and it becomes a different size than the original, the card will not work. This is usually always the case since the tools used always resulted in a recompiled card that was smaller in size than originally started with. The e-Reader was developed in collaboration between Nintendo, Creatures, and HAL Laboratories, but it's thought the compression techniques for the peripheral were developed by HAL directly. This is because the `vpk0` compression format present in the e-Reader cards are also present in several HAL N64 games, including Super Smash Bros. We do know that HAL used two encoding algorithms in their `vpk0` compression: LSZZ and Huffman Encoding. However, we do not know the exact settings they used for the e-Reader so I had to find a workaround in order to replicate their compression process.

This meant I needed to modify the header data to accurately flag the VPK size and the general size of the card.



```
          00 01  02 03  04 05  06 07   08 09  0a 0b  0c 0d  0e 0f
00000000  00 30  01 02  00 01  08 10   00 00  10 12  e0 41  01 00   .0..........àA..
00000010  68 08  10 2c  eb 19  00 00   00 08  4e 49  4e 54  45 4e   h..,ë.....NINTEN
00000020  44 4f  00 22  00 09  22 ba   0b 02  00 00  00 00  50 7c   DO."..".......P|
00000030  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000040  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000050  00 41  6e 69  6d 61  6c 20   43 72  6f 73  73 69  6e 67   .Animal Crossing
00000060  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00000070  00 00  db 05  .. ..  .. ..   .. ..  .. ..  .. ..  .. ..   ..Û.
```

Revisiting this diagram, the VPK size is denoted by the final two bytes of the header data. Note that these are written in little endian, so the VPK size here is 0x05DB in hexadecimal or 1499 bytes in decimal. Thus, when recompiling a new VPK I had to make note of the size and change this value to the proper hexadecimal declaring the new size of the VPK. If this value is off by a single digit, the card will not read. Further, there is also a general size info which also keeps track of VPK size. In green, the hexadecimal 0x020BBA22 is actually 32-bits of information written in binary, with bits 9-23 repeating the VPK size plus an additional 2 bytes for where the size is declared. With this information we can do some bit math to properly rewrite the size info flag to accurately represent newly recompressed data.



Example of the hex/bit math required to recreate the general size flag in the data header

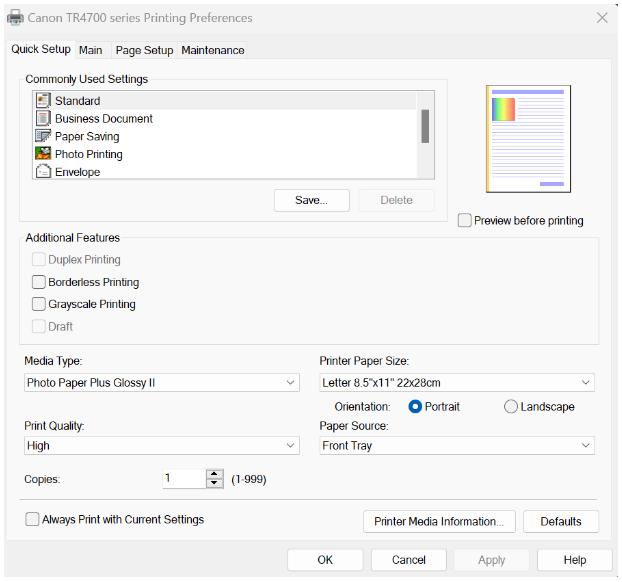
With this formula, I was able to recreate the general size flag within the header data to accurately represent any newly compressed data I introduced into the code. There is still a problem however, as there is a global check to ensure that the total size of the e-Reader card data is 2112 bytes. Since the compression brings us below this limit, we simply fill the data with `00` or `ff` as blank "padding" data to ensure the card size is correct. On top of this, there are still two checksum flags in the header that need to be corrected. Mathematically, these checksums are the complement of the sum of all halfwords in all data fragments. This can be done manually, but luckily user BlackShark from Project Pokémon created a program to correct these checksums automatically. This was originally created for use with the Pokémon e-Reader cards, but luckily this code can also be repurposed here. Big thanks to BlackShark for also posting about the above math and formulas which made the process easier.

With all of these programs and complications settled, we can now successfully decompile and recompile e-Reader data accurately. So I decompiled one of the Animal Crossing e-Reader cards, rewrote the bytes responsible for generating the item attached to the letter to include the *Super Mario Bros* item code, and recompiled through the process above. From this, I was able to generate a working RAW file that was properly scanned in Dolphin emulator and gave me a working *Super Mario Bros* furniture item in-game. From here, it was time to generate and print a physical dot code based off this RAW file and test on console.

Printing the Dot Codes

To actually print the dot codes physically, you will need access to a modern printer with at least 600x600dpi printing resolution and high quality glossy photo paper. Converting the RAW files into printable dot codes can be challenging, but luckily Tim Schuerewegen developed a tool to easily print them back in 2004 that still works today. You can get the e-Reader dot code printing tool from CaitSith's site here. Most modern inkjet/laser printers are capable of printing accurate dot codes, but this may be trial-and-error depending on the type of printer you have. Following this video tutorial may also prove helpful.

Here are my Canon TR4722 printer settings for reference:



Canon TR4700 series Printing Preferences

Quick Setup | Main | Page Setup | Maintenance

Commonly Used Settings: [✓ Standard] / Business Document / Paper Saving / Photo Printing / Envelope

Save... | Delete

☐ Preview before printing

Additional Features
☐ Duplex Printing
☐ Borderless Printing
☐ Grayscale Printing
☐ Draft

Media Type: Photo Paper Plus Glossy II

Printer Paper Size: Letter 8.5"x11" 22x28cm

Orientation: [✓ Portrait] Landscape

Print Quality: High

Paper Source: Front Tray

Copies: 1 (1-999)

☐ Always Print with Current Settings

Printer Media Information... | Defaults

OK | Cancel | Apply | Help

Downloads

You can download the files for the ***Super Mario Bros*** e-Reader card here:

.raw	(for printing dot codes and use on emulator)
.bin	(for modifying)
.vpk	(chunked)

You can download the files for ***The Legend of Zelda*** e-Reader card here:

.raw	(for printing dot codes and use on emulator)
.bin	(for modifying)
.vpk	(chunked)

Acknowledgements

This project could not have been accomplished by myself, and I’d like to thank several people for posting their findings on the internet for me to use as research:

- **CaitSith2** and **Tim Schuerewegen** for developing and compiling the e-Reader dev tools from 2004-2007.
- **RollingStar** (GitHub) / **do_0m** (GameFAQs) for starting the Python project of automating the decompilation/recompilation process.
- **BlackShark** for shedding light on e-Reader header data and creating the program to automatically fix header checksums.
- **Martin Korth** for documenting extracted e-Reader development information and publishing them on his site.
- **tehzz** for shedding light on the vpk0 compression format and its history.
- **Anzomia** for creating the YouTube tutorial on how to print dot codes, sparking my interest in the project.

References

Below are several links I used as research to make this project possible:

- https://www.caitsith2.com/ereader/index.htm
- https://crates.io/crates/vpk0
- https://problemkaputt.de/gbatek-gba-cart-e-reader-data-format.htm
- https://problemkaputt.de/gbatek-gba-cart-e-reader-program-code.htm
- https://projectpokemon.org/home/forums/topic/58692-tutorial-creating-custom-trainerberry-e-cards/
- https://web.archive.org/web/20191102214417/http://users.skynet.be/firefly/gba/e-reader/